EVIDENCE®
EMBEDDING TECHNOLOGY

MAGNETI MARELLI

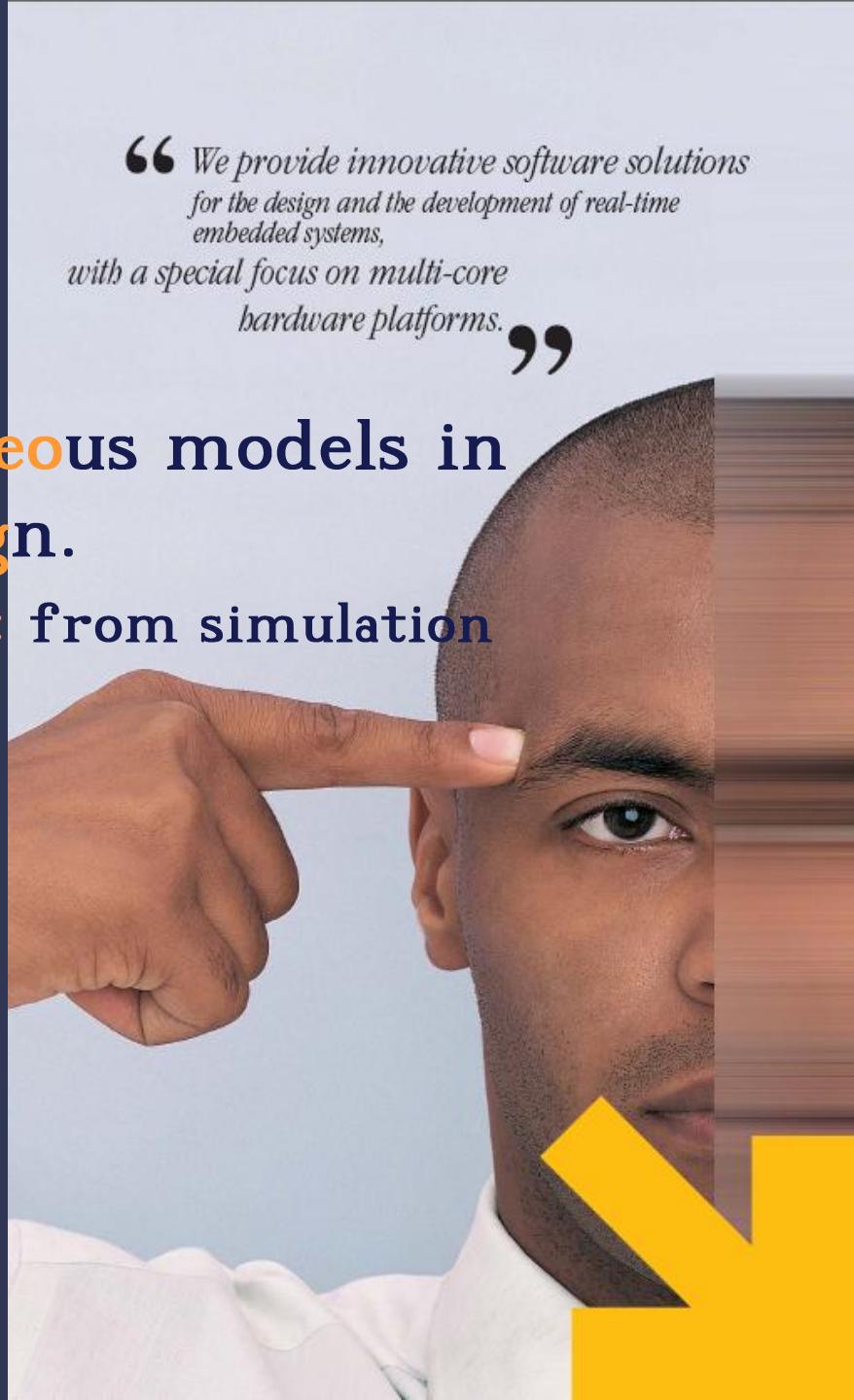
Methods and tool for the modeling and timing analysis of embedded systems

Integration of heterogeneous models in automotive systems design.

Keeping model views consistent from simulation and analysis to code generation.

Paolo Gai, Marco Di Natale, Giacomo Gentile, Nicola Ariotti

Evidence Srl, Ghezzano (PI),
Scuola Superiore S. Anna, Pisa,
Magneti Marelli, Bologna - Italy



INTERESTED

Via Carducci 56, 56010 S.Giuliano Terme Pisa - Italy

Web: http://www.evidence.eu.com

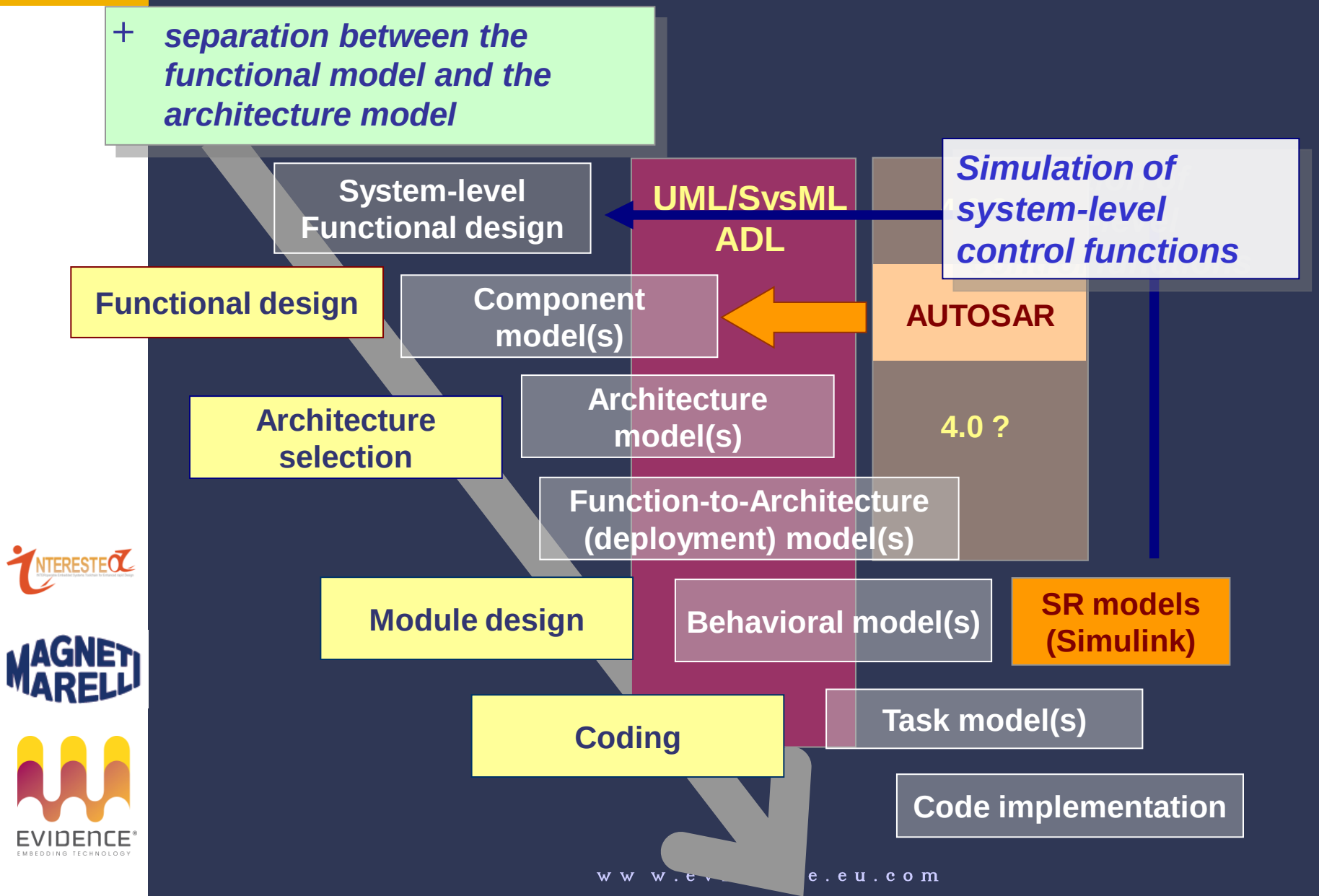
Outline

- The development of automotive embedded systems: a realistic flow with methods, models and standards
- Gaps and Points of friction in the integration of heterogeneous models
  - Interoperability of models: UML/SysML – AUTOSAR and Simulink (SR)
  - From functional models to task models: semantics preservation issues, optimization opportunities
- The role of timing analysis in the development of automotive systems
  - Modeling time constraints and attributes: the MARTE profile in SysML, opportunities and challenges in AUTOSAR
  - Tool integration for timing analysis
- Keeping AUTOSAR models aligned for Simulation, timing analysis and code generation
  - The Magneti Marelli fuel injection case in INTERESTED

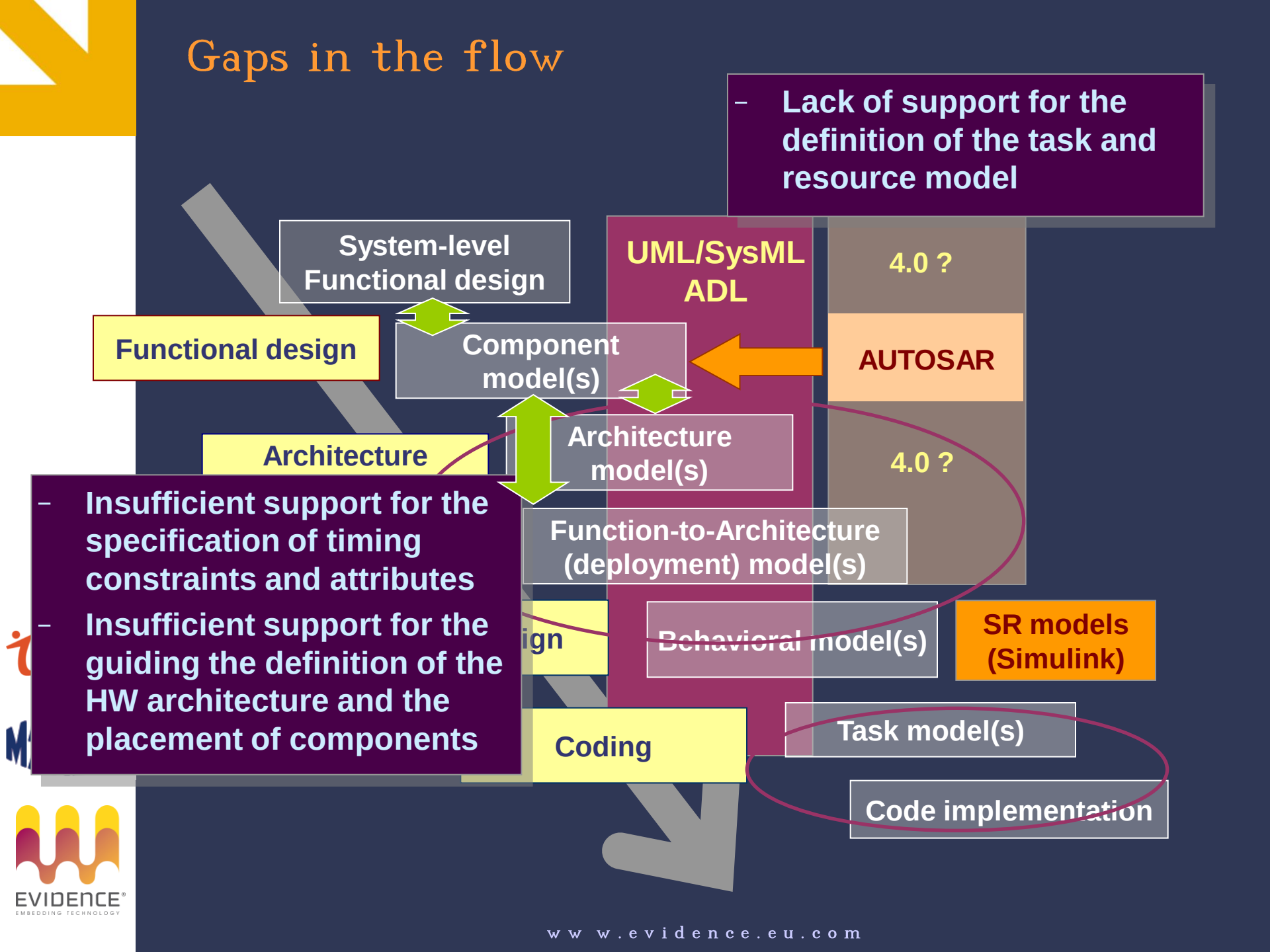
Motivation: automotive systems sample reqmts

- Several applications (an example is fuel injection control) are developed with the following characteristics:
- Very cost-sensitive
  - Faster CPU often not an option, limited availability of memory
- Large number of functional subsystems (~200) interacting exchanging several hundreds of communication signals
- System is multirate and characterized by tight timing constraints
- Utilization is very high (>90%) and mode changes are required
- Model-based design using Simulink (SR) models
- Need to plan transition to AUTOSAR-based development
- Need for:
  - Supporting the design of the task and resource model starting from system-level models to Simulink and AUTOSAR models
    - Enforce semantics preservation (partial orders, comm flows)
    - Optimize wrt performance metrics within timing constraints
  - Algorithms that optimize memory use while enforcing deadlines
    - Use of preemption thresholds
  - Model and control mode changes

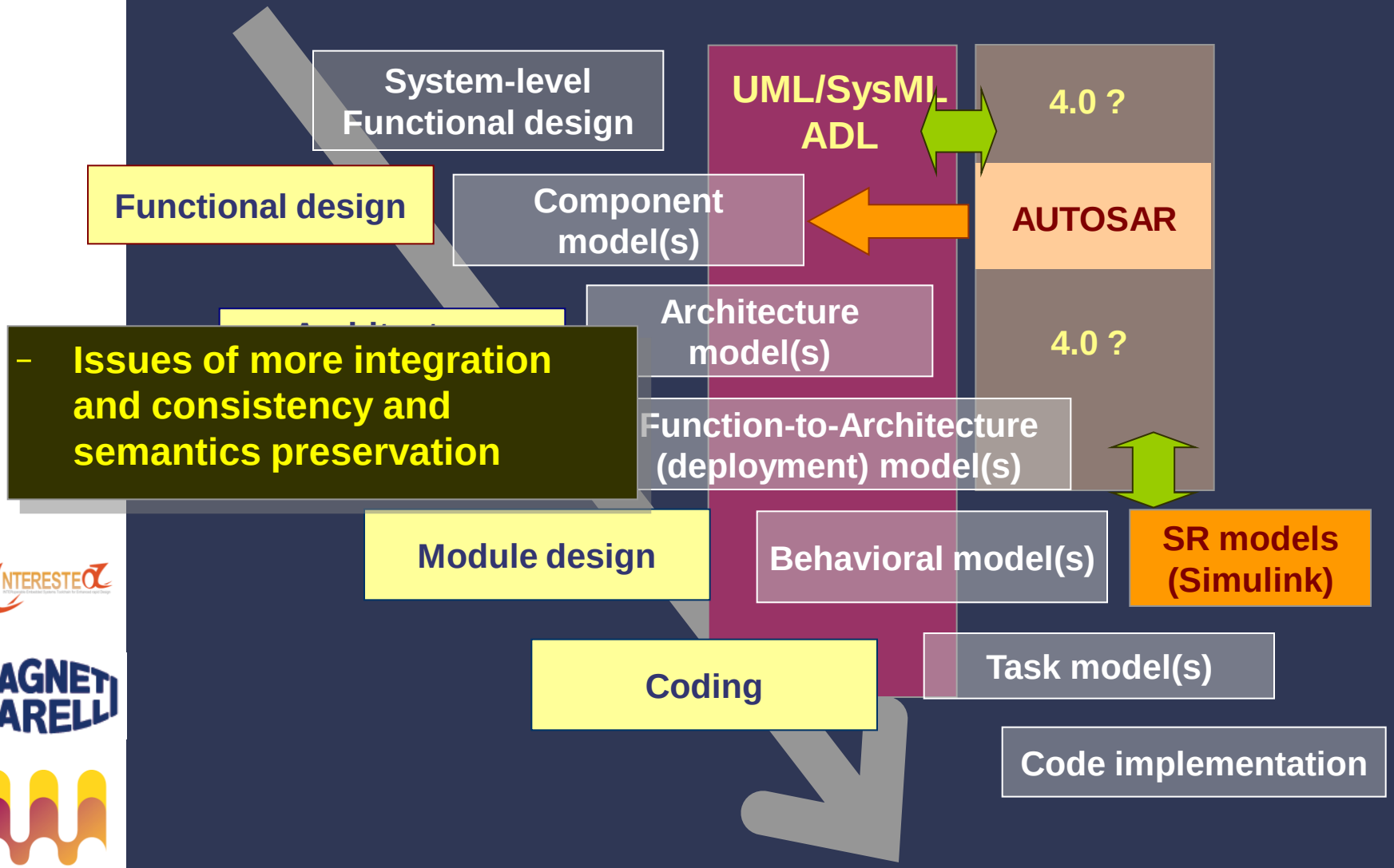
A Typical flow: Heterogeneous models



Gaps in the flow

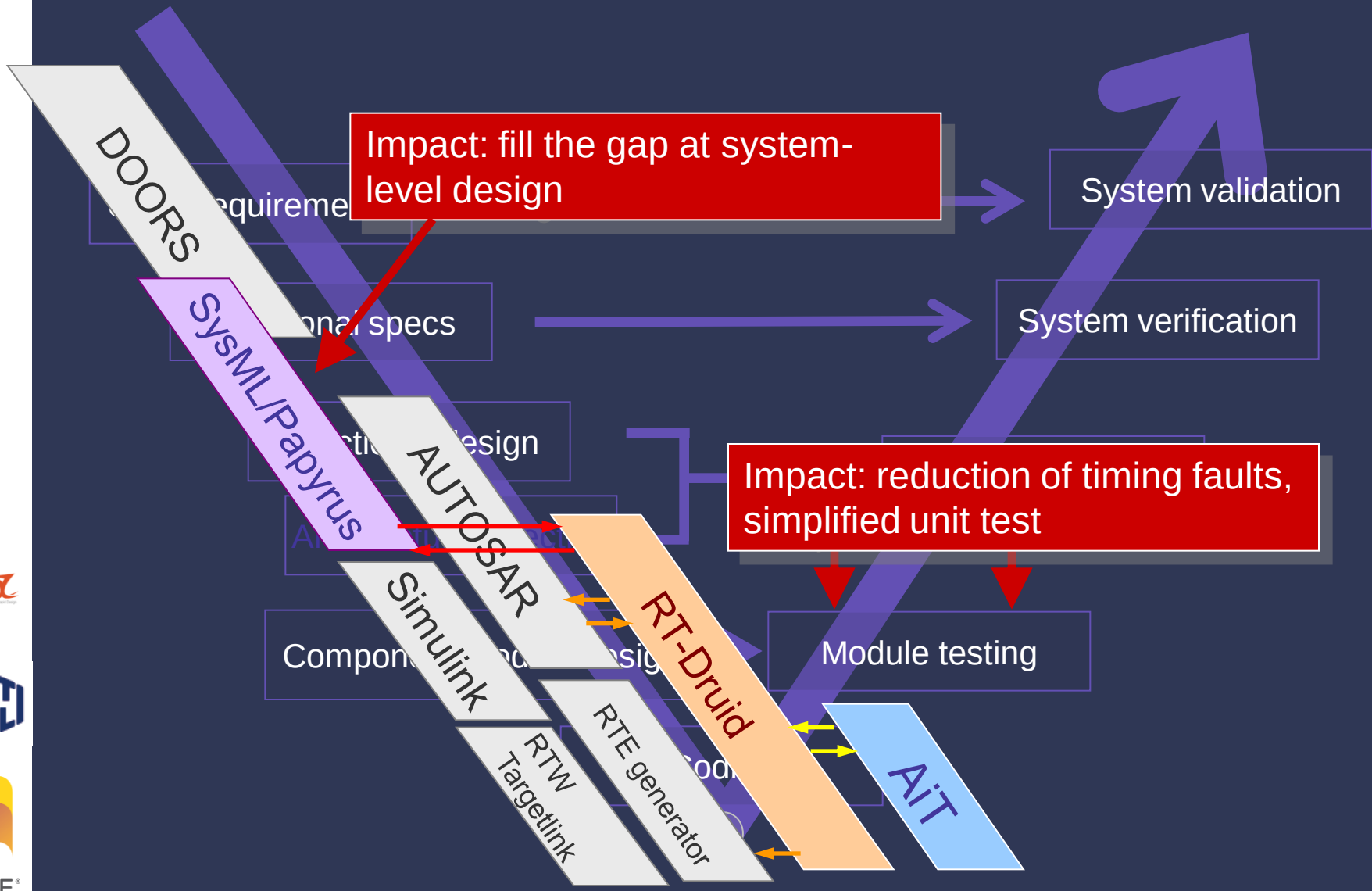


Frictions in the integration



- **Issues of more integration and consistency and semantics preservation**

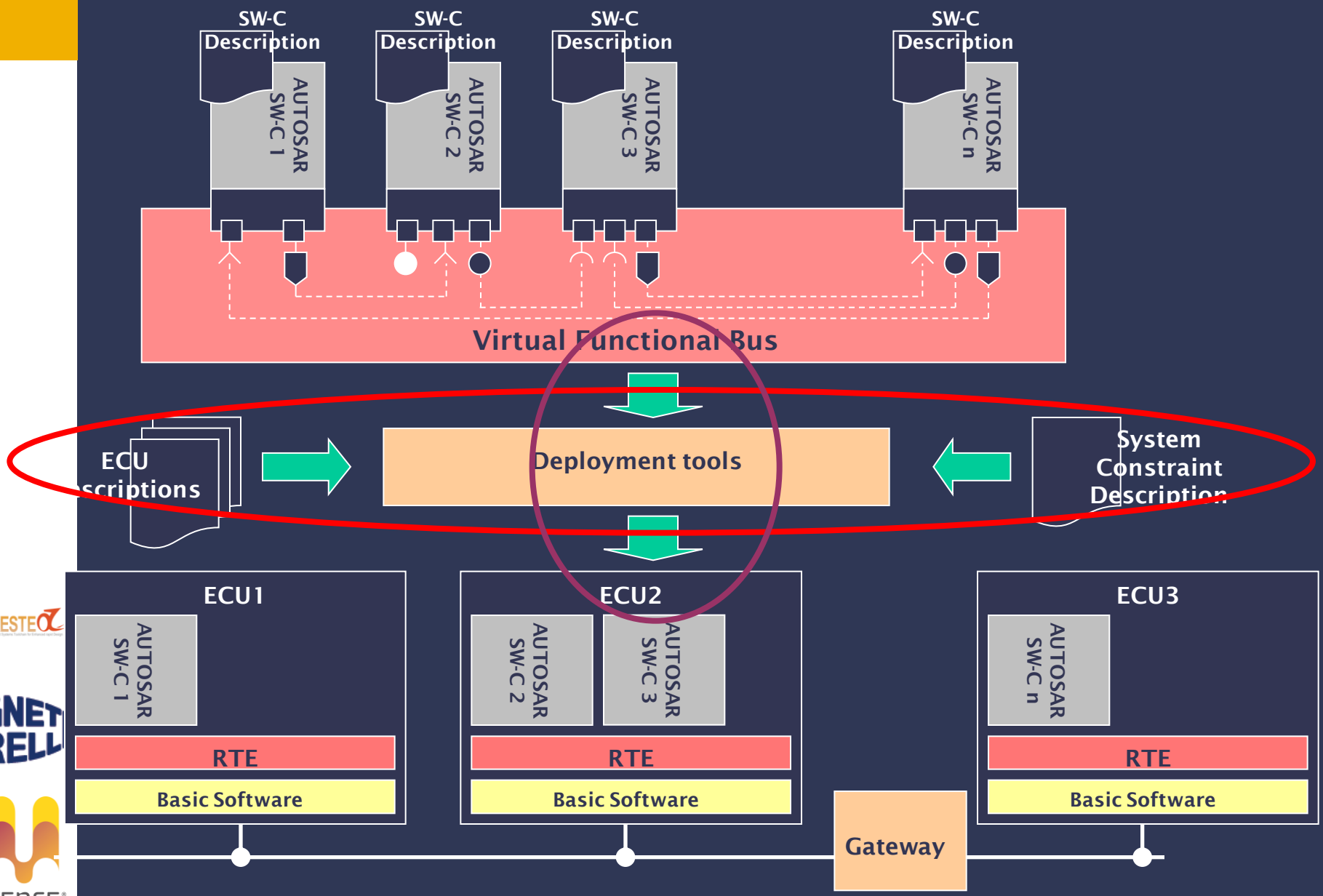
Heterogeneous models in INTERESTED



Process Gaps — SysML/MARTE-AUTOSAR-SR models

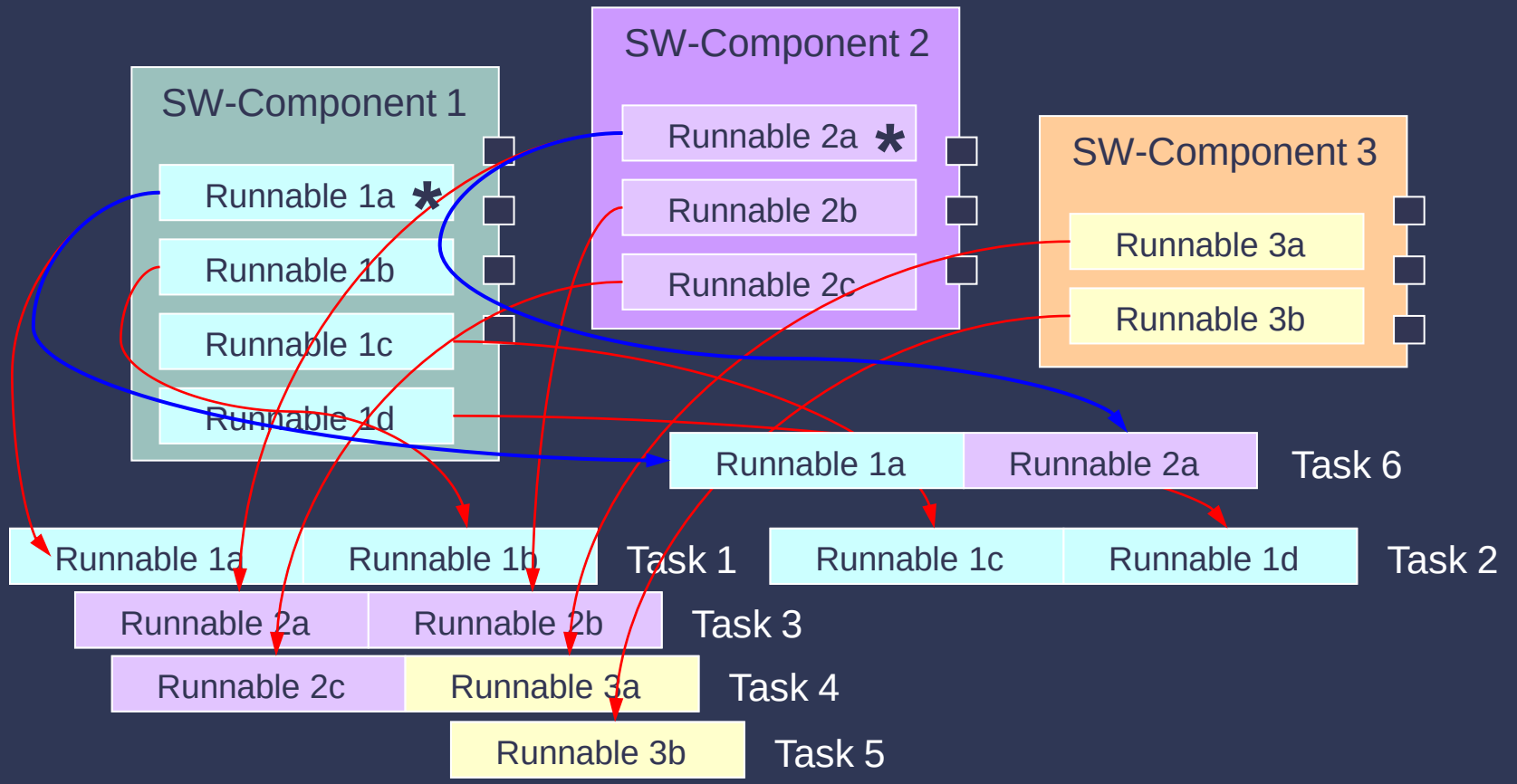
- SysML/MARTE
  - Providing the definition of the (hardware) execution platform
  - Adding an explicit task model
  - Providing the definition of the mapping of the behavior (functions/methods) to tasks
  - Providing the mapping of signals into messages
  - Providing the definition of the mapping of tasks into ECUs and messages into buses
  - Providing the semantics to timed events (far from ideal)
- Goals
  - Define mapping/translation rules from/to AUTOSAR (largely completed)
  - Define mapping/translation rules to/from SR models
  - Define mapping to time analysis viewpoint

AUTOSAR Architecture



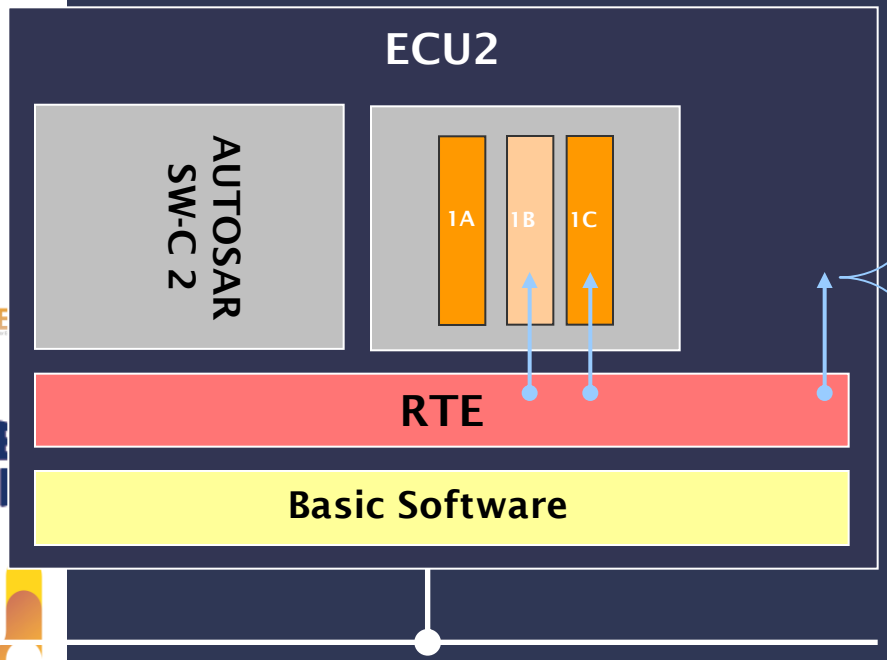
AUTOSAR behavior: Runnable entities

Runnable entities define the behavior of the components.
They are the smallest code-fragments that are provided by the component and are (at least indirectly) a subject for scheduling by the operating system. .



AUTOSAR runnables and the RTE

- The activation model and the synchronization in the execution of runnables (local and remote) is specified in a middle-level layer, the RTE (Run-Time Environment), which is the runtime implementation of the VFE
- The RTE layer **is local to each ECU** and responsible for triggering the execution of runnables using the following events



- ***Timing Event***
  (periodic execution).
- ***DataReceivedEvent***
  (reception on Sender/Receiver)
- ***OperationInvokedEvent***
  (invocation of Client/Server)
- ***DataSendCompleteEvent***
  (sending on Sender/Receiver)
- ***WaitPoint***
  (blocks a runnable waiting for an Event)

Advanced features of timing analysis in INTERESTED

- Most tools for schedulability analysis ***assume availability of the task model*** and require that each task is characterized as being periodic or sporadic
  - with a period or minimum interarrival T_i, a worst case exec. time C_i, a deadline D_i and possibly an activation offset O_i.
- However, in a model-based design flow such a task model is typically not available or hidden inside the mechanisms of the code generation tool, or produced late in the development flow.
- In INTERESTED, the RT-Druid tool starts from the system description at the functional level
  - AUTOSAR component and runnables or
  - SysML blocks - UML subsystems or composite objects or a
  - Simulink subsystem.

Advanced features of timing analysis in INTERESTED

- RT-Druid requires the definition of
  - The communication, synchronization and activation model of the functional blocks, including activation events, precedence constraints and the definition of the data flows.
  - In addition, each functional block is characterized by its worst-case execution time and its worst-case memory

UML/MARTE is currently the only means to obtain a description of the time constraints in the same context of the objects/events to which they apply.

(from UML or AUTOSAR tools), and supports the user in

- the definition of the mapping of functional blocks to tasks,
- the assignment of priorities to tasks,
- the definition of preemption groups for allowing minimum use of stack memory within the time constraints.
- the selection of the communication primitives
- *and enforces preservation of semantics rules from functional models*

Advanced features of timing analysis in INTERESTED

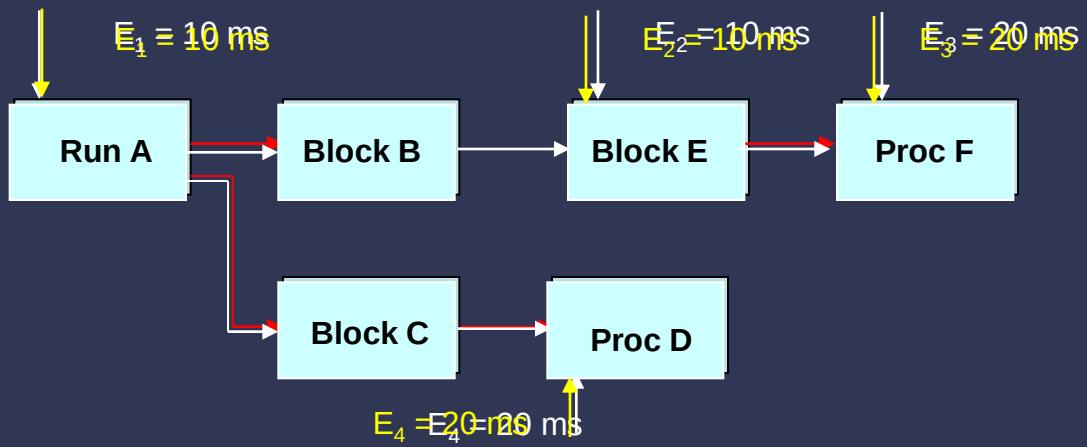
- The issues that need to be considered in this case are the following (the AUTOSAR runnable is used as an example of functional block):
  - The assignment of runnables to tasks must be performed in such a way that it is consistent with the definition of the activation events.
  - For example, a task with period T_i can only provide an implementation to periodic runnables with period kT_i, with an activation semantics of type 1-every-k, or activated directly with client-server semantics by a runnable with period T_i.
  - The execution order of the runnables inside the task, the priority, and possibly the activation offset assignment should be defined in accordance with the order of execution (precedence constraints) imposed on the runnables.

Advanced features of timing analysis in INTERESTED

- The issues that need to be considered in this case are …
  - The worst-case execution time of a task is obtained from the execution times of the blocks mapped into it. This can be trivial in some cases. However, execution patterns of type 1-every-k may require modeling a task with multiple virtual tasks or more sophisticated task models, such as the multiframe model.
  - The assignment of priorities and preemption thresholds can be performed to minimize the memory required for the stack space while satisfying the time constraints.
  - The tool will also support the designer in the selection of the communication mechanisms among blocks with a library of possible options that include shared variables protected by disabling preemption, priority ceiling semaphores or wait free mechanisms and will also provide an estimate on cases when preemption among blocks/runnables is impossible given the current estimates of the WCETs.

Optimal Synthesis of Task and comm. Model

Q: what is the best runnable-to-task mapping?



Run A	Run E	Run B	Run F(*)	Task 1 - 10 ms	Proc A	Proc E	Proc B
Proc W				Task 2 - 40 ms	Proc W		
Proc C				Task 3 - 10 ms	Proc C		
Proc D				Task 4 - 20 ms	Proc D	Proc F	

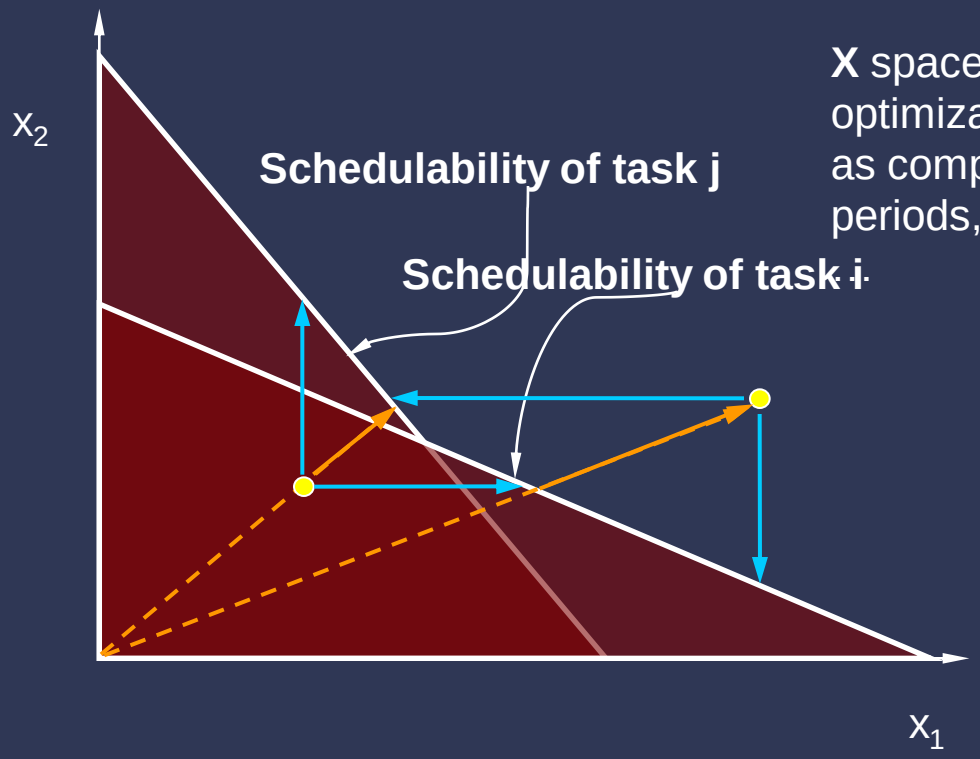
Pro: No need to protect communication between E and F.
Cons: Less scheduling flexibility, limited priority inversion

Advanced features of timing analysis in INTERESTED

- The tool will also …
- estimate the memory required by the communication variables given the selection of the communication mechanisms.
- attempt at performing optimization on the selection of the communication mechanisms.
- signal when preservation of the communication semantics is guaranteed and when it is violated by the task implementation.

Advanced features of timing analysis in INTERESTED

- The tool not only provides the feasibility test and the worst case response times of tasks and blocks, but also sensitivity analysis results .



X space of design optimization variables, such as computation times, periods, placement, priorities

Alignment of models

- In our flow, AUTOSAR models have three objectives:
  - simulate the functional behavior,
  - provide information to RTDruid for performing worst-case timing analysis and,
  - generate the code for the implementation of the communication and synchronization on a given basic software. This code will have to integrate with the code generated from Simulink models.
- The use of AUTOSAR for the purposes of modeling, simulation and code generation has several issues,
- The situation can only partly benefit from the notion of time and timed events as introduced in the latest (4.0) AUTOSAR release
- The timed event model is at best cumbersome if the purpose is,
  - (for example) to build a common discrete time base the integration of Simulink models.
  - In addition, the use of timed events is currently not supported by tools.

Alignment of models

- In a fuel injection control system several functions are executed at a given rotation angle of the engine shaft. A crankshaft position sensor provides a reference *sporadic* hardware signal.
- This signal triggers a set of functions, typically mapped for execution into a single task.
- *In AUTOSAR, a sporadic signal of this type cannot be directly defined*. However, inside the application model, the runnables that are executed in response to the crankshaft position signal need to be activated in response to an RTE event, (this is the only legal way for a runnable to be activated in AUTOSAR!).
- One possible solution is the following.
  - A basic SW runnable interfaces the hardware signal coming from the crankshaft position sensor to the application tasks.
  - This BSW runnable writes into a send/receive port to forward to the application the sensor position signal.
  - The software components containing application runnables activated in response to this signal will define a matching port. Inside them, the runnables are defined to be activated on the event of data received on the port.

Alignment of models

- The basic SW component is the main variation point with respect to the three model versions.
- For simulation, the BSW runnable is activated periodically by a Counter/Alarm pair. Inside, it contains the simulation stub that generates the stream of the crankshaft sensor signals.These signals are represented by writes into the send/receive port on which the application runnables are waiting.
- For code generation, the basic SW component is replaced by a device driver that will explicitly call the RTE API function for writing into the data port at the end of its execution.
- For worst case timing analysis, it is necessary to have information about the worst-case arrival rate of the activation events for the sporadic task into which all runnables are mapped. In AUTOSAR 4.0 this indication could come directly from a Timed Event triggering information. In current AUTOSAR releases the sporadic rate must be deduced from the rate of the writes into the corresponding data port, which would require navigating the communication model graph to the basic SW runnable and the corresponding Alarm (which however, can only be periodic !)

Multicore Extension

- Project extension: development flow aimed at multicore platforms
- Issues:
  - Operating system support including memory protection, multicore extensions to OSEK resources
  - Time predictability against remote blocking on shared resources.
  - AUTOSAR support for multicore
    - Definition of task placement and OS configuration
    - Timing analysis on AUTOSAR models
  - Extension of flow-preserving communication mechanisms (Rate Transition blocks) to multicores

Conclusion

- The development of complex automotive applications in a model-based flow must face the problem of integrating heterogeneous models
  - At the very least AUTOSAR and Simulink
  - The connection is well supported for “code plumbing”, much less for correct semantic integration
  - The mapping of the functional model into the execution platform model and the definition of the task model is still a manual task
  - Timing support in AUTOSAR (v 4.0) is still incomplete and cumbersome
    - Only developed for worst-case timing analysis, not simulation
    - Lack of a formal timed event model